Optimisation de la méthode de rejet de la signature Wave

Étienne Burle, Nicolas Sendrier, Thomas Debris-Alazard, Equipe COSMIQ, Inria Paris

20/08/2020

Le contexte général

La cryptographie dite asymétrique ou à clef publique permet de mettre en place des algorithmes de chiffrement et de signature. Elle est toujours basée sur un problème difficile. Aujourd'hui, les algorithmes de chiffrement et de signature sont essentiellement basés sur le problème du logarithme discret et sur la factorisation de grands nombres. Cependant, la sécurité de ces algorithmes est menacée par l'ordinateur quantique.

L'idée d'un ordinateur quantique est apparue dans les années 70-80 et de premiers prototypes ont été proposé dans les années 1990. En 1994, Shor trouve un nouvel algorithme (l'algorithme de Shor). C'est un algorithme quantique (donc exécutable par un ordinateur quantique) qui permet de factoriser des grands nombres en un temps polynomial en la taille des entrées. Il y a donc mise en danger de la sécurité de nombreux algorithmes cryptographiques actuels, ceci d'autant plus que la recherche n'a cessé de progresser depuis pour fabriquer des ordinateurs quantiques de taille raisonnable. Certes, ceux dont on dispose actuellement sont encore trop petits pour pouvoir menacer les chiffrements actuels, mais de nouvelles avancées sont annoncées régulièrement, par exemple par Google[AAB$^+$19] en septembre dernier. De nombreuses entreprises comme IBM ou Google, mais aussi les États-Unis ou la Chine investissent dans l'ordinateur quantique.

Il y a donc nécessité de renouveler les algorithmes cryptographiques à clef publique. Déjà pour être en mesure de chiffrer et signer les messages futurs le jour où l'ordinateur quantique tiendra ses promesses, mais aussi pour assurer que certains messages actuels puissent rester indéchiffrables pour les 50 prochaines années. C'est dans cette optique que l'agence américaine des standards National Institute of Standards and Technology (NIST) a lancé en décembre 2016 un processus international en vue de standardiser des primitives de chiffrement et de signature électronique [1]. La cryptographie ayant pour but de résister à l'ordinateur quantique s'appelle la cryptographie post-quantique. Les outils mathématiques étudiés pour mettre ne place ces nouveaux algorithmes sont principalement les réseaux euclidiens et les codes correcteurs d'erreur, mais aussi les fonctions de hachage, les polynômes multi-variés ou les isogénies. Ce stage porte sur la cryptographie basée sur les codes correcteurs d'erreur.

Le premier à avoir eu l'idée d'utiliser cet outil est McEliece en 1978[McE78]. Il se basait sur la difficulté pour décoder un code linéaire quelconque. Il y a depuis eu de nombreux schémas de chiffrement basés sur les codes, cependant aucune signature à base de code efficace n'a pu être trouvée. Mais en décembre 2019 a été proposée la signature *Wave*[DST19], une nouvelle signature cryptographique basée sur les codes. Je l'ai étudiée et tenté d'améliorer sa mise en œuvre pendant mon stage.

Le problème étudié

J'ai travaillé sur la signature *Wave*. Lorsqu'on signe un message m, il y a plusieurs signatures s possibles. Le signataire possède un secret qui lui donne une des signatures s possibles. Le choix de cette signature peut permettre à un éventuel attaquant d'obtenir de l'information sur le secret en faisant des études statistiques. Jusqu'ici, la fuite d'information était évitée grâce à une "méthode de rejet". Mon travail a consisté à optimiser une distribution de probabilité discrète, afin de pouvoir diminuer l'utilisation de cette méthode de rejet, voire la supprimer, sans pour autant laisser fuiter de l'information. Cela permettrait d'alléger la mise en œuvre de la signature, ainsi que d'éviter des attaques dites par "canaux auxiliaires". La signature étant nouvelle, j'ai été le premier à me pencher sur cette question. L'intérêt

1. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`

était d'améliorer *Wave*, qui pourrait par la suite devenir une option sérieuse de signature à base de code correcteur d'erreur.

La contribution proposée

J'ai d'abord regardé comment diminuer le nombre de rejet effectué, en faisant varier par tâtonnement la distribution de probabilité. Après avoir constaté de cette manière qu'il était possible de faire diminuer l'utilisation de la méthode de rejet beaucoup plus que ce qui avait déjà été fait, j'ai recherché des méthodes plus théoriques pour améliorer la signature.

J'ai d'abord tenté d'utiliser les transformées de Fourrier. Malheureusement, cela n'a pas donné de résultat. J'ai ensuite utilisé des méthodes d'optimisation quadratique pour optimiser la distribution de probabilité. D'abord une version approchée de l'algorithme du gradient sous contrainte, puis un algorithme appelé algorithme d'Uzawa. Ces deux algorithmes ont donné des résultats similaires, diminuant de beaucoup l'utilisation de la méthode de rejet, mais pas suffisamment pour pouvoir s'en passer.

Je me suis également intéressé à un problème asymptotique qui n'avait pas encore été résolu. Il s'agissait de voir, lorsqu'on fait tendre la taille des paramètres de *Wave* vers l'infini, s'il était possible d'avoir une utilisation de la méthode de rejet polynomiale en la taille des paramètres. Jusqu'à aujourd'hui, la méthode de rejet a été montrée efficace seulement pour des paramètres fixés. Cela permettait de savoir s'il est possible d'augmenter la taille des paramètres sans voir l'utilisation du rejet grandir trop vite. J'ai pu répondre par l'affirmative au problème, sous la réserve d'une conjecture calculatoire vérifiée numériquement, en exhibant une suite de distributions de probabilités qui convenait.

Les arguments en faveur de sa validité

Pour ce qui est du travail à paramètres fixé, tout a été vérifié numériquement, en *Sage* surtout, et en partie en *C*. La plupart des résultats ont été vérifié en *Sage* avec des paramètres plus petits que ceux utilisés réellement, afin de diminuer les temps de calculs, mais les meilleurs résultats (ceux obtenus par optimisation quadratique), ont été vérifié en *C* pour des paramètres standards d'utilisation. Les résultats obtenus sont donc certains pour des paramètres standards, mais il n'est pas certains que les méthodes utilisées fonctionneraient en cas de changement de paramètres.

Quant au résultat asymptotique, il s'agit d'une démonstration théorique, qui s'appuie cependant sur une conjecture. Il s'agit d'une conjecture calculatoire (s'assurer de la stricte négativité de deux fonctions), qui a été vérifiée numériquement en *Sage* pour des paramètres asymptotiques standards. Peut-être qu'un changement de paramètres asymptotiques démentirait cette conjecture, mais il n'y a a priori pas de raison de les changer. Ce résultat est donc très fiable.

La bilan et les perspectives

J'ai réussi à beaucoup diminuer l'utilisation de la méthode de rejet, mais pas encore suffisamment pour pouvoir s'en passer. Il est probablement possible d'encore améliorer les résultats, notamment en faisant des hypothèses a priori sur la distribution de probabilité qui convient, facilitant l'utilisation des algorithmes d'utilisation quadratique. Par ailleurs, peut-être que continuer de chercher des méthodes d'analyse harmonique, avec les transformées de Fourrier, permettrait de d'obtenir de meilleurs méthodes. Mais je n'ai pour l'instant pas réussi à trouver dans ce domaine de méthode prenant en compte les contraintes des distributions de probabilités (positivité des termes, dont la somme est égale à 1).

Peut-être sera-t-il possible en continuant à chercher dans ce domaine de se passer de la méthode de rejet. Il y a de nombreuses questions encore ouvertes sur *Wave*.

Notations

Généralités La notation $x \triangleq y$ signifie que x est défini comme étant égale à y. $\mathbb{F}_q$ est la notation pour le corps fini à q éléments. Pour deux entiers a et b tels que $a \leq b$, l'ensemble des entiers $\{a, a+1, \cdots, b\}$ est désigné par $[\![a, b]\!]$. Pour un ensemble $\mathcal{E}$, $|\mathcal{E}|$ désigne son cardinal.

Notations matrices, vecteurs Les vecteurs seront notés avec des lettres grasses minuscules (telles que $\mathbf{e}$) et les matrices avec des majuscules (telles que $\mathbf{H}$). Les vecteurs seront en *notation ligne* et nous noterons $\mathbb{F}_q^{m \times n}$ l'ensemble des matrices à coefficient dans $\mathbb{F}_q$ ayant m lignes et n colonnes.

Pour un vecteur $\mathbf{x}$, sa i-ème composante sera notée x_i ou $x(i)$. De même pour une matrice $\mathbf{H}$, sa composante de coordonnée (i, j) sera désignée par $h_{i,j}$. On notera $\mathbf{x}_{\mathcal{I}}$ le vecteur dont les coordonnées sont celles de $\mathbf{x} = (x_i)_{1 \leq i \leq n}$ mais restreintes à l'ensemble $\mathcal{I} \subseteq [\![1, n]\!]$, *i.e :* $\mathbf{x}_{\mathcal{I}} = (x_i)_{i \in \mathcal{I}}$. De même, $\mathbf{H}_{\mathcal{I}}$ désignera la matrice dont les colonnes sont celles de $\mathbf{H} \in \mathbb{F}_q^{m \times n}$ restreintes à $\mathcal{I}$, *i.e :* $\mathbf{H}_{\mathcal{I}} = (h_{i,j})_{1 \leq i \leq m; j \in \mathcal{I}}$

Le support d'un vecteur $\mathbf{x}$ de longueur n est quant à lui défini comme

$$\mathrm{Supp}(\mathbf{x}) \triangleq \{1 \leq i \leq n : x_i \neq 0\}$$

et le poids de Hamming de $\mathbf{x} \in \mathbb{F}_q^n$ est défini comme :

$$|\mathbf{x}| \triangleq |\mathrm{Supp}(\mathbf{x})|$$

Notations probabilistes Soit S un ensemble fini, alors $x \hookleftarrow S$ dénote le fait que x est tiré uniformément dans S. De même, pour une variable aléatoire X ou une distribution $\mathcal{D}$, la notation $x \hookleftarrow X$ (resp. $x \hookleftarrow \mathcal{D}$) dénote le fait que X est tirée selon la variable aléatoire X (resp. la distribution $\mathcal{D}$). Pour deux variables aléatoires X et Y, $X \sim Y$ signifiera que X et Y sont équi-distribués. La même notation sera utilisée pour une variable aléatoire X et une distribution $\mathcal{D}$, la notation $X \sim \mathcal{D}$ signifiant que X est distribuée selon $\mathcal{D}$.

Si on souhaite expliciter l'espace probabiliste sur lequel les probabilités ou espérances sont considérées, il sera indiqué en indice de la probabilité la variable aléatoire ou distribution sur laquelle cette dernière est calculée. Par exemple, si $\mathcal{D}$ désigne une distribution et $\mathbf{x} \hookleftarrow \mathcal{D}$, la probabilité de l'événement "$\mathbf{x} \in \mathcal{E}$" sera noté $\mathbb{P}_{\mathbf{x}}(\mathbf{x} \in \mathcal{E})$.

Distance statistique La distance statistique entre deux distributions discrètes $\mathcal{D}_0$ et $\mathcal{D}_1$ sur un même espace $\mathcal{E}$ est définie comme :

$$\rho(\mathcal{D}_0, \mathcal{D}_1) \triangleq \frac{1}{2} \sum_{x \in \mathcal{E}} |\mathcal{D}_0(x) - \mathcal{D}_1(x)|$$

Pour deux variables aléatoires X et Y à valeur dans le même espace, nous noterons aussi $\rho(X, Y)$ la distance statistique entre la distribution $\mathcal{D}_X$ de X et la distribution $\mathcal{D}_Y$ de Y : $\rho(X, Y) \triangleq \rho(\mathcal{D}_X, \mathcal{D}_Y)$.

Notations asymptotiques Soient deux fonctions $f, g : \mathbb{N} \to \mathbb{R}$. On notera :

$$f(n) \triangleq O\left(g(n)\right) \iff \exists M > 0, \forall n \in \mathbb{N} : |f(n)| \leq M \cdot |g(n)|$$

$$f(n) \triangleq \widetilde{O}\left(g(n)\right) \iff \exists k \in \mathbb{N}, f(n) = \log^k |g(n)| \cdot O\left(g(n)\right)$$

Formule préliminaire Soit q un entier. L'entropie q-aire est définie sur $[0, 1]$ comme :

$$h_q(x) \triangleq -(1-x)\log_q(1-x) - x \log_q\left(\frac{x}{q-1}\right).$$

Alors on a pour des entiers q, n et $w = \omega n$ avec $\omega \in]0, 1[$, asymptotiquement en n :

$$\binom{n}{w}(q-1)^w \sim \frac{1}{\sqrt{2\pi n \omega(1-\omega)}} q^{n h_q(\omega)} \tag{1}$$

1 La signature *Wave*

Wave est une nouvelle signature cryptographique basée sur les codes proposée en décembre dernier [DST19], dont je vais décrire le fonctionnement ici. Je commencerai par rappeler le principe d'une signature cryptographique et des codes correcteurs d'erreurs, puis je détaillerai le coeur de la signature *Wave*, enfin j'expliquerai la méthode de rejet ainsi que l'objet de mon stage.

1.1 Signatures et codes correcteurs d'erreur

1.1.1 Signature cryptographique

Généralités Une signature à clef publique sert à authentifier l'auteur d'un message et à vérifier que le message n'a pas été altéré. En voici une définition formelle :

Définition 1. *Une signature à clef publique est donnée par un triplet d'algorithmes (*`KeyGen`*,* `Sgn`*,* `Vrfy`*) et un couple de fonctions* $(m(\lambda), n(\lambda))$ *polynomiales en le paramètre de sécurité* λ *tels que :*

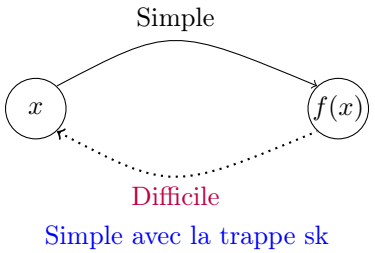
- `KeyGen`*,* la génération de clef*, est un algorithme probabiliste polynomial d'entrée* 1^λ *et renvoyant une paire de clefs publique et secrète* $(\mathrm{pk}, \mathrm{sk})$*,*
- `Sgn`*,* la signature*, est un algorithme probabiliste polynomial prenant en entrée* sk*, un message* $\mathbf{m} \in \{0,1\}^{m(\lambda)}$*, et renvoyant une signature* $\mathbf{y} \in \{0,1\}^{n(\lambda)}$*,*
- `Vrfy`*, est un algorithme à valeur dans* $\{0,1\}$ *et prenant comme entrée* pk*,* $\mathbf{m} \in \{0,1\}^{m(\lambda)}$ *et un élément de* $\{0,1\}^{n(\lambda)}$ *tels que :*

$$\mathtt{Vrfy}^{\mathrm{pk}}(\mathbf{m}, \mathtt{Sgn}(\mathbf{m})) = 1$$

Il existe plusieurs façons de concevoir des signatures, mais *Wave* est une signature de type hache et signe.

Signature de type hache et signe On dispose de $f : \mathcal{A} \longrightarrow \mathcal{D}$, une fonction à sens unique à trappe, de trappe sk, c'est-à-dire :

- pour toute entrée $x \in \mathcal{A}$, $f(x)$ se calcule facilement,
- aucun algorithme ne connaissant pas sk ne peut trouver un élément de $f^{-1}(f(x))$ en temps polynomial en moyenne,
- on calcule facilement un élément de $f^{-1}(f(x))$ avec sk.



Soit $\mathcal{H}$ une fonction de hachage cryptographique. La signature d'un message $\mathbf{m}$ et sa vérification consiste simplement en :

- Signature : $\sigma \in f^{-1}(\mathcal{H}(\mathbf{m}))$ calculée grâce à sk.
- Vérification : $f(\sigma) = \mathcal{H}(\mathbf{m})$

Fonctionnement de la signature Alice veut envoyer un message $\mathbf{m}$ à Bob, qui doit pouvoir authentifier Alice comme étant l'auteur de ce message, et vérifier qu'il n'a pas été altéré. Alice est la seule qui connaît la trappe sk, donc à pouvoir inverser f. Elle calcule donc $\sigma = f^{-1}(\mathcal{H}(\mathbf{m}))$, puis envoie le couple $(\mathbf{m}, \sigma)$ à Bob. Celui-ci calcule alors $\mathcal{H}(\mathbf{m})$ et $f(\sigma)$. Si les deux valeurs sont égales, le message provient bien d'Alice puisqu'elle est la seule à pouvoir calculer σ. De même le message $\mathbf{m}$ n'a pas été altéré en $\mathbf{m}'$ sinon $f(\sigma) = \mathcal{H}(\mathbf{m})$ n'aurait pas la même valeur que $\mathcal{H}(\mathbf{m}')$. On remarque que f doit être *surjective*.

Nous allons maintenant voir comment utiliser les codes correcteurs d'erreurs pour construire une signature de type hache et signe.

1.1.2 Théorie des codes

Un code linéaire $\mathcal{C}$ de longueur n et de dimension k est un sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k. Un tel code est caractérisé par une matrice de parité $\mathbf{H}$, qui est une matrice de dimensions $(n-k)\times n$ de rang plein telle que $\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_q^n : \mathbf{c}\mathbf{H}^\intercal = \mathbf{0}\}$. Pour $\mathbf{x} \in \mathbb{F}_q^n$, $\mathbf{s} = \mathbf{x}\mathbf{H}^\intercal$ est appelé *syndrome* de $\mathbf{x}$, et vaut $\mathbf{0}$ si et seulement si $\mathbf{x} \in \mathcal{C}$

Tout code $\mathcal{C}$ étant par définition un sous-espace vectoriel de $\mathbb{F}_q^n$, il possède un ensemble $\mathcal{I} \subseteq [\![1, n]\!]$, dit *ensemble d'information*, de taille k déterminant uniquement tout mot du code :

$$\forall \mathbf{x} \in \mathbb{F}_q^k,\ \exists! \mathbf{c} \in \mathcal{C} :\ \mathbf{c}_{\mathcal{I}} = \mathbf{x}$$

Le problème classique utilisé en cryptographie basée sur les codes est le suivant :

Problème 1. (Décodage générique du syndrome).

Entrée : $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_q^{n-k}$, $w \in [\![0, n]\!]$
Problème : Trouver $\mathbf{x} \in \mathbb{F}_q^n$, $|\mathbf{x}| = w$, *tel que* $\mathbf{x}\mathbf{H}^\intercal = \mathbf{s}$

C'est à partir de ce problème que nous allons construire la signature à sens unique à trappe de *Wave*.

1.2 Fonction à sens unique et décodage

Le problème du décodage générique du syndrome mène naturellement à poser la fonction suivante :

$$\begin{array}{rccc} f_{w,\mathbf{H}} : & \{\mathbf{y} \in \mathbb{F}_q^n : |\mathbf{y}| = w\} & \longrightarrow & \mathbb{F}_q^{n-k} \\ & \mathbf{x} & \longmapsto & \mathbf{x}\mathbf{H}^\intercal = \mathbf{s} \end{array}$$

où $\mathbf{H}$ est une matrice de parité d'un code $\mathcal{C}$. Inverser $f_{w,\mathbf{H}}$ nécessite de résoudre une instance du problème de décodage générique du syndrome.

Remarque importante Nous supposerons toujours que le syndrome $\mathbf{s}$ est uniformément distribué dans $\mathbb{F}_q^{n-k}$ lorsqu'on cherchera à inverser $f_{w,\mathbf{H}}$. En effet, on a vu lors de la description des signatures cryptographiques que l'on cherche à trouver l'antécédent du haché d'un message $\mathcal{H}(\mathbf{m})$, où $\mathcal{H}$ est une fonction de hachage cryptographique. On supposera cette fonction de hachage comme aléatoire uniforme, selon le modèle classique de l'oracle aléatoire. C'est le modèle utilisé pour prouver la sécurité de *Wave*, voir [Deb19, 5].

Nous allons d'abord décrire l'algorithme de Prange qui permet en moyenne de résoudre ce problème en temps polynomial pour certains w, puis nous présenterons la trappe associée à $f_{w,\mathbf{H}}$ dans *Wave*. Cette trappe permet d'inverser la fonction pour des valeurs de w telles que sans la trappe l'inversion est difficile.

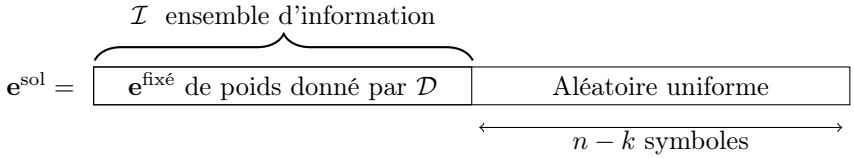
1.2.1 Algorithme de Prange

Nous présentons une version généralisée de l'algorithme de Prange originel, issue de [Deb19, 3.1].

Cet algorithme cherche à résoudre le problème de décodage générique du syndrome pour une matrice de parité $\mathbf{H}$ *quelconque*. Fixons $(\mathbf{H}, \mathbf{s}) \in \mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$, w un entier, ainsi qu'un vecteur $\mathbf{e}^{\text{fixé}} \in \mathbb{F}_q^k$. On va trouver un ensemble d'information $\mathcal{I}$, puis déterminer l'unique vecteur $\mathbf{e}^{\text{sol}} \in \mathbb{F}_q^n$ tel que $\mathbf{e}^{\text{sol}}_{\mathcal{I}} = \mathbf{e}^{\text{fixé}}$ et $\mathbf{e}^{\text{sol}}\mathbf{H}^\intercal = \mathbf{s}$. Le poids de $\mathbf{e}^{\text{fixé}}$ est déterminé par une distribution $\mathcal{D}$.

Cela revient à résoudre un système à $n-k$ équations et n inconnues, où on choisit arbitrairement la valeur de k inconnues. Le fait que ces k inconnues soient choisies selon l'ensemble d'information $\mathcal{I}$ garantit que le système obtenu à $n-k$ équations et $n-k$ inconnues est bien résoluble. Les $n-k$ coordonnées obtenues en dehors de l'ensemble d'information pourront être considérées comme choisies uniformément dans le cas du syndrome uniforme, comme nous le verrons dans la Proposition 1.

Cette approche est résumée dans le schéma suivant :



Une itération de l'algorithme est donnée dans l'Algorithme 1. On itère le nombre de fois nécessaire pour obtenir une solution de poids w.

Algorithme 1 IterationPrangeGen$(\mathbf{H}, \mathbf{s}, \mathcal{D})$— Une itération de l'algorithme de Prange

Paramètre : q, n, k, $\mathcal{D}$ une distribution de probabilités sur $[\![0, k]\!]$

Require: $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_q^{n-k}$
Ensure: $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$
1: $t \hookleftarrow \mathcal{D}$
2: $\mathcal{I} \leftarrow$ InfoSet$(\mathbf{H})$ ▷ InfoSet$(\mathbf{H})$ *retourne un ensemble d'information du code de matrice de parité* $\mathbf{H}$
3: $\mathbf{x} \hookleftarrow \{\mathbf{y} \in \mathbb{F}_q^n \mid |\mathbf{y}_{\mathcal{I}}| = t\}$
4: $\mathbf{e} \leftarrow$ PrangeAlgèbreLinéaire$(\mathbf{H}, \mathbf{s}, \mathcal{I}, \mathbf{x})$
5: **return** $\mathbf{e}$

function PrangeAlgèbreLinéaire$(\mathbf{H}, \mathbf{s}, \mathcal{I}, \mathbf{x})$

Require: $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_q^{n-k}$, $\mathcal{I}$ un ensemble d'information du code de matrice de parité $\mathbf{H}$, $\mathbf{x} \in \mathbb{F}_q^n$
Ensure: $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$ et $\mathbf{e}_{\mathcal{I}} = \mathbf{x}_{\mathcal{I}}$
$\mathbf{P} \leftarrow$ une $n \times n$ permutation renvoyant $\mathcal{I}$ sur les k dernières coordonnées
$(\mathbf{A} \mid \mathbf{B}) \leftarrow \mathbf{H}\mathbf{P}$ ▷ $\mathbf{A} \in \mathbb{F}_q^{(n-k)\times(n-k)}$
$(\mathbf{0} \mid \mathbf{e}') \leftarrow \mathbf{x}$ ▷ $\mathbf{e}' \in \mathbb{F}_q^k$
$\mathbf{e} \leftarrow \left((\mathbf{s} - \mathbf{e}'\mathbf{B}^\intercal)\left(\mathbf{A}^{-1}\right)^\intercal, \mathbf{e}'\right)\mathbf{P}^\intercal$
return $\mathbf{e}$

Proposition 1. *Pour* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ *fixé et* $\mathbf{s}$ *uniformément distribué sur* $\mathbb{F}_q^{n-k}$*, nous avons pour la sortie* $\mathbf{e}$ *de* IterationPrangeGen$(\mathbf{H}, \mathbf{s})$ *:*

$$|\mathbf{e}| = S + T$$

où $S \in [\![0, n-k]\!]$ *et* $T \in [\![0, k]\!]$ *sont des variables aléatoire indépendantes. Ici* S *désigne le poids de Hamming d'un vecteur uniformément distribué sur* $\mathbb{F}_q^{n-k}$ *tandis que* $\mathbb{P}(T = t) = \mathcal{D}(t)$*. La distribution* $|\mathbf{e}|$ *est alors donnée par :*

$$\mathbb{P}_{\mathbf{s},c}(|\mathbf{e}| = w) = \sum_{t=0}^{w} \frac{\binom{n-k}{w-t}(q-1)^{w-t}}{q^{n-k}}\mathcal{D}(t), \quad \mathbb{E}_{\mathbf{s},c}(|\mathbf{e}|) = \overline{\mathcal{D}} + \frac{q-1}{q}(n-k)$$

où la probabilité est calculé sur $\mathbf{s}$ *uniforme, l'aléa interne* c *de l'algorithme et* $\overline{\mathcal{D}} = \sum_{t=0}^{k} t\mathcal{D}(t)$*.*

Preuve de la Proposition 1. Par définition toute sortie de l'algorithme 1 s'écrit comme :

$$\left((\mathbf{s} - \mathbf{e}'\mathbf{B}^\intercal)\left(\mathbf{A}^{-1}\right)^\intercal, \mathbf{e}'\right)\mathbf{P}^\intercal$$

où $\mathbf{P}$ est une matrice de permutation. De plus, la matrice $\mathbf{A}$ étant une bijection, la distribution du poids de ce vecteur est identique à celle de $|(\mathbf{s} - \mathbf{e}'\mathbf{B}, \mathbf{e}')|$. D'où

$$\mathbb{P}_{\mathbf{s},c}(|\mathbf{e}| = w) = \sum_{t=0}^{w} \mathbb{P}_{\mathbf{s},c}(|(\mathbf{s} - \mathbf{e}'\mathbf{B}, \mathbf{e}')| = w - t \mid |\mathbf{e}'| = t)\mathbb{P}(|\mathbf{e}'| = t)$$

$$= \sum_{t=0}^{w} \mathbb{P}_{\mathbf{s},c}(|(\mathbf{s} - \mathbf{e}'\mathbf{B}, \mathbf{e}')| = w - t \mid |\mathbf{e}'| = t)\mathcal{D}(t)$$

Maintenant, $\mathbf{s} \in \mathbb{F}_q^{n-k}$ étant uniformément distribué, il en est de même de $\mathbf{s} - \mathbf{e}'\mathbf{B}$ quelque soient $\mathbf{e}'$ et $\mathbf{B}$ qui sont indépendants de $\mathbf{s}$. Donc nous avons :

$$\mathbb{P}_{\mathbf{s},c}(|(\mathbf{s} - \mathbf{e}'\mathbf{B}, \mathbf{e}')| = w - t \mid |\mathbf{e}'| = t) = \frac{\binom{n-k}{w-t}(q-1)^{w-t}}{q^{n-k}}$$

d'où le résultat. □

L'algorithme généralisé de Prange consiste à répéter IterationPrangeGen($\mathbf{H}, \mathbf{s}, \mathcal{D}$) jusqu'à obtenir le poids w souhaité. Le choix de $\mathcal{D}$ va dépendre du w voulu. $\mathcal{D}$ va prendre des valeurs faibles si on souhaite un poids faible, et inversement. Plus précisément, $\frac{q-1}{q}(n-k)$ étant le poids moyen des coordonnées en dehors de l'ensemble d'information :

— si $w < \frac{q-1}{q}(n-k)$, $\mathcal{D}$ est choisie comme étant la distribution nulle,
— si $w \in \left[\frac{q-1}{q}(n-k), k + \frac{q-1}{q}(n-k)\right]$, $\mathcal{D}$ est la distribution constante égale à $w - \frac{q-1}{q}(n-k)$,
— si $w > k + \frac{q-1}{q}(n-k)$, $\mathcal{D}$ est choisie comme étant la distribution égale à k.

La probabilité sur $(\mathbf{H}, \mathbf{s})$ de succès de l'algorithme 1 (c'est-à-dire d'obtenir le bon poids w) en paramétrant $\mathcal{D}$ comme ci-dessus est de :

$$\widetilde{O}\left(\frac{\binom{n-k}{w-j}(q-1)^{w-j}}{\min\left(\binom{n}{w}(q-1)^w, q^{n-k}\right)}\right)$$

$$\text{où} \begin{cases} j = 0 & \text{si } w \leq \frac{q-1}{q}(n-k) \\ j = k & \text{si } w \geq k + \frac{q-1}{q}(n-k) \\ j = w - \frac{q-1}{q}(n-k) & \textit{sinon} \end{cases}$$

Voir [Deb19, 2.3, 3.1.2] pour la preuve. Ceci amène aux conclusions suivantes :

Poids w atteignables Supposons $\frac{w}{n}$ et $R \triangleq \frac{k}{n}$ fixés.

* Si $\frac{w}{n} \in \left[\frac{q-1}{q}(1-R), R + \frac{q-1}{q}(1-R)\right]$, alors on peut résoudre avec l'algorithme de Prange le décodage en temps $P(n)(1+o(1))$ pour un certain polynôme P.
* Si $\frac{w}{n} \notin \left[\frac{q-1}{q}(1-R), R + \frac{q-1}{q}(1-R)\right]$, la complexité de Prange est exponentielle, i.e. : $2^{\alpha n(1+o(1))}$, pour un certain $\alpha > 0$.

Les différentes améliorations de l'algorithme de Prange ainsi que les différents algorithmes mis au point pour résoudre le problème du syndrome [LB88, Ste88, Dum91, MMT11, BJMM12, MO15, BM17] sont de même nature en terme de complexité asymptotique, c'est-à-dire que l'on reste polynomial dans l'intervalle précédent et exponentiel en dehors. La complexité moyenne du décodage semble donc être bien mesurée par l'algorithme de Prange.

1.2.2 Trappe et décodage

Notons $w^-_{\text{Facile}} \triangleq \frac{q-1}{q}(n-k)$ et $w^+_{\text{Facile}} \triangleq k + \frac{q-1}{q}(n-k)$.

Nous avons vu que grâce à l'algorithme de Prange, on pouvait inverser $f_{w,\mathbf{H}}$ facilement en moyenne pour $w \in [\![w^-_{\text{Facile}}, w^+_{\text{Facile}}]\!]$. La fonction est difficile à inverser pour w en dehors de cette zone pour une matrice $\mathbf{H}$ quelconque. Mais il existe une trappe permet au signataire de l'inverser facilement en dehors de cet intervalle. La trappe consiste à choisir la matrice $\mathbf{H}$ de façon particulière. Plus précisément, $\mathbf{H}$ est une matrice de parité obtenue d'un code $(U, U+V)$, comme défini ci-dessous.

Définition 2. *Un code $\mathcal{C}$ du type $(U, U+V)$ de longueur n et de dimension k est un code construit à partir de deux autres codes U et V de longueurs $n/2$, de dimensions k_U et k_V, tel que $k = k_U + k_V$ et défini comme :*

$$\mathcal{C} = \{(\mathbf{u} \mid \mathbf{u} + \mathbf{v}); \mathbf{u} \in U, \mathbf{v} \in V\}$$

En notant $\mathbf{H}_U$ *et* $\mathbf{H}_V$ *les matrices de parités respectives des codes* U *et* V*, ainsi que* $\mathbf{H}_{UV}$ *celle du code* $\mathcal{C}$*, on a :*

$$\mathbf{H}_{UV} = \left(\begin{array}{c|c} \mathbf{H}_U & 0 \\ \hline -\mathbf{H}_V & \mathbf{H}_V \end{array}\right)$$

Notation importante Pour tout $\mathbf{e} \in \mathbb{F}_q^n$, nous noterons $(\mathbf{e}_U, \mathbf{e}_V) \in (\mathbb{F}_q^{n/2})^2$ l'unique couple tel que

$$\mathbf{e} = (\mathbf{e}_U \mid \mathbf{e}_U + \mathbf{e}_V)$$

Remarque importante Dans [DST19] et [Deb19, 5.1], *Wave* est construit grâce à des codes du type $(U, U+V)$ *généralisés*. Pour simplifier l'exposé, nous nous limitons à la définition simple ci-dessus.

Fonction à sens unique à trappe de *Wave* La trappe consiste à choisir une matrice de parité $\mathbf{H}$ particulière. On choisit deux codes U et V quelconques de longueurs n et de dimensions respectives k_U et k_V. On note $\mathbf{H}_{UV}$ la matrice de parité du code $(U, U+V)$ correspondant, définie comme précédemment. On choisit aléatoirement $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ inversible et $\mathbf{P} \in \mathbb{F}_q^{n\times n}$ une matrice de permutation, et on prend $\mathbf{H} = \mathbf{S}\mathbf{H}_{UV}\mathbf{P}$. On peut donc considérer la fonction à sens unique à trappe suivante :

$$\begin{array}{rccc} f_{w,\mathbf{H}} : & \{\mathbf{y} \in \mathbb{F}_q^n : |\mathbf{y}| = w\} & \longrightarrow & \mathbb{F}_q^{n-k} \\ & \mathbf{x} & \longmapsto & \mathbf{x}\mathbf{H}^\intercal = \mathbf{s} \end{array}$$

La clef secrète sk est l'ensemble $\{\mathbf{S}, \mathbf{P}, \mathbf{H}_U, \mathbf{H}_V\}$, $\mathbf{H}_U$ et $\mathbf{H}_V$ étant les matrices de parités respectives de U et V. La trappe suppose difficile de distinguer $\mathbf{H}$ d'une matrice de parité d'un code quelconque.

Décodage Voyons comment inverser $f_{w,\mathbf{H}}$ en connaissant la clef secrète.

Soit $\mathbf{s} \in \mathbb{F}_q^{n-k}$. On cherche $\mathbf{e} \in \mathbb{F}_q^n$ tel que $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$.

$$\begin{aligned} \mathbf{e}\mathbf{H}^\intercal = \mathbf{s} &\iff \mathbf{e}\mathbf{P}^\intercal{\mathbf{H}_{UV}}^\intercal\mathbf{S}^\intercal = \mathbf{s} \\ &\iff (\mathbf{e}\mathbf{P}^\intercal){\mathbf{H}_{UV}}^\intercal = \mathbf{s}(\mathbf{S}^\intercal)^{-1} \end{aligned}$$

$\mathbf{P}$ et $\mathbf{S}$ étant connues du détenteur du secret, cela revient, pour $\mathbf{s} \in \mathbb{F}_q^{n-k}$, à chercher $\mathbf{e} \in \mathbb{F}_q^n$ tel que $\mathbf{e}{\mathbf{H}_{UV}}^\intercal = \mathbf{s}$. En notant $\mathbf{s} = (\mathbf{s}^U \mid \mathbf{s}^V)$ avec $\mathbf{s}^U \in \mathbb{F}_q^{n/2-k_U}$ et $\mathbf{s}^V \in \mathbb{F}_q^{n/2-k_V}$, on a :

$$\begin{aligned} \mathbf{e}{\mathbf{H}_{UV}}^\intercal = \mathbf{s} &\iff (\mathbf{e}_U \mid \mathbf{e}_U + \mathbf{e}_V)\left(\begin{array}{c|c} {\mathbf{H}_U}^\intercal & -{\mathbf{H}_V}^\intercal \\ \hline 0 & {\mathbf{H}_V}^\intercal \end{array}\right) = (\mathbf{s}^U \mid \mathbf{s}^V) \\ &\iff \mathbf{e}_U{\mathbf{H}_U}^\intercal = \mathbf{s}^U \quad \text{et} \quad -\mathbf{e}_U{\mathbf{H}_V}^\intercal + (\mathbf{e}_U + \mathbf{e}_V){\mathbf{H}_V}^\intercal = \mathbf{s}^V \\ &\iff \mathbf{e}_U{\mathbf{H}_U}^\intercal = \mathbf{s}^U \quad \text{et} \quad \mathbf{e}_V{\mathbf{H}_V}^\intercal = \mathbf{s}^V \end{aligned}$$

Ainsi, il suffit pour décoder de trouver $\mathbf{e}_U$ et $\mathbf{e}_V$ tels que $\mathbf{e}_U{\mathbf{H}_U}^\intercal = \mathbf{s}^U$ et $\mathbf{e}_V{\mathbf{H}_V}^\intercal = \mathbf{s}^V$.

U et V étant a priori des codes quelconques, on pourrait utiliser Prange d'abord sur le code U, puis sur V pour inverser $f_{w,\mathbf{H}}$. Mais ceci serait possible en temps polynomial uniquement pour des w appartenant à l'intervalle $[\![w^-_{\text{Facile}}, w^+_{\text{Facile}}]\!]$. Pour en sortir, on va d'abord décoder V grâce à l'algorithme de Prange, puis décoder U en utilisant Prange de manière à utiliser de façon avantageuse la structure en $(U, U+V)$ du code.

Obtenir des solutions de gros poids avec des codes $(U, U+V)$ Une fois le vecteur $\mathbf{e}_V$ obtenu grâce à Prange, on choisit $\mathbf{e}_U(i)$ sur k_U positions données par un ensemble d'information comme :

$$\begin{cases} \mathbf{e}_U(i) \neq 0 \\ \mathbf{e}_U(i) + \mathbf{e}_V(i) \neq 0 \end{cases}$$

Cela est possible, quelque soit $\mathbf{e}_V$, si q est choisi ≥ 3. C'est le choix qui sera fait dans *Wave*. Ainsi, une fois le second Prange terminé, on obtient d'après la Proposition 1 une solution $\mathbf{e}$ de longueur n telle que :

— $2k_U$ coordonnées sont $\neq 0$.

— les $n - 2k_U$ autres coordonnées sont uniformément distribuées sur $\mathbb{F}_q$.

Le poids attendu de la solution est alors donné par :

$$\mathbb{E}(|\mathbf{e}|) = 2k_U + \frac{q-1}{q}(n-2k_U) = \frac{q-1}{q}n + \frac{2k_U}{q} \triangleq w^+_{UV}$$

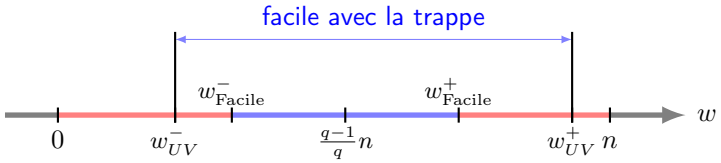
Ainsi, il suffit d'avoir $k_U > k_V$, pour que $2k_U > k$ et donc $w^+_{UV} > w^+_{\text{Facile}}$.

Solutions de petit poids Il existe une technique analogue pour obtenir des solutions de petits poids, on obtient

$$w^-_{UV} \triangleq \begin{cases} \frac{q-1}{q}(n-2k) & \text{si } k \leq \frac{n}{2q} \\ \frac{2(q-1)^2}{(2q-1)q}(n-k) & \text{sinon} \end{cases}$$

Dans *Wave*, on s'intéresse au décodage à poids fort plutôt qu'à poids faible.

Tout ceci est résumé dans le schéma suivant :



On prendra désormais $\boxed{q=3}$.

L'algorithme formel de décodage est décrit dans l'Algorithme 2. On remarque, pour le décodage de U, l'utilisation d'une famille de distribution $\mathcal{D}^t_U$ paramétrée par t le poids de $\mathbf{e}_V$, résultat du décodage de V. Cette distribution détermine, lorsqu'on choisit $\mathcal{I}$ l'ensemble d'information de U lors du décodage de U, le cardinal de $\{i \in \mathcal{I} : \mathbf{e}_V(i) \neq 0\}$. Ceci a une importance cruciale pour obtenir le poids w voulu.

Algorithme 2 DécodageUVSimple($\mathbf{H}_U, \mathbf{H}_V, \mathbf{s}, w, \mathcal{D}_V, \mathcal{D}^t_U$)

Paramètre : $k_U, k_V, n, \mathcal{D}_V$ une distribution de probabilités sur $[\![0, k_V]\!]$, $\mathcal{D}^t_U$ une famille de distributions de probabilités sur $[\![\min(0, t + k_U - n/2), t]\!]$ paramétrées par $t \in [\![0, n/2]\!]$.

1: $\mathbf{e}_V \leftarrow$ DécodageV($\mathbf{H}_V, \mathbf{s}^V, \mathcal{D}_V$)
2: $t \leftarrow |\mathbf{e}_V|$
3: $k_{\neq 0} \hookleftarrow \mathcal{D}^t_U$
4: $\mathbf{e}_U \leftarrow$ DécodageU($\mathbf{H}_U, \mathbf{s}^U, \mathbf{e}_V, k_{\neq 0}, w$)
5: $\mathbf{e} \leftarrow (\mathbf{e}_U \mid \mathbf{e}_U + \mathbf{e}_V)$
6: **return** $\mathbf{e}$

function DécodageV($\mathbf{H}, \mathbf{s}, \mathcal{D}$)

  $\mathbf{e}_V \leftarrow$ IterationPrangeGen($\mathbf{H}, \mathbf{s}, \mathcal{D}$)
  return $\mathbf{e}_V$

function DécodageU($\mathbf{H}, \mathbf{s}, \mathbf{y}, k_{\neq 0}, w$)

  repeat
    $\mathcal{I} \leftarrow$ InfoSetPW($\mathbf{H}, \mathbf{y}, k_{\neq 0}$) ▷ InfoSetPW*($\mathbf{H}, \mathbf{x}, k_{\neq 0}$) retourne un ensemble d'information du code de matrice de parité* $\mathbf{H}$ *tel que* $|\{i \in \mathcal{I} : \mathbf{x}_i \neq 0\}| = k_{\neq 0}$
    $\mathbf{x}_U \hookleftarrow \{\mathbf{x} \in \mathbb{F}_3^{n/2} \mid \forall i \in \mathcal{I},\ \mathbf{x}_i \notin \{0, -\mathbf{y}_i\}$ et $\text{Supp}(\mathbf{x}) \subseteq \mathcal{I}\}$
    $\mathbf{e}_U \leftarrow$ PrangeAlgèbreLinéaire($\mathbf{H}, \mathbf{s}, \mathcal{I}, \mathbf{x}_U$)
  until $|(\mathbf{e}_U \mid \mathbf{e}_U + \mathbf{e}_V)| = w$
  return $\mathbf{e}_U$

1.3 Méthode de rejet et objectif du stage

Cependant, l'algorithme de décodage permettant d'inverser $f_{w,\mathbf{H}}$ décrit précédemment est sujet aux fuites d'information. En effet, chaque $\mathbf{s} \in \mathbb{F}_3^{n-k}$ possède plusieurs antécédents par $f_{w,\mathbf{H}}$. Un attaquant, en collectionnant les signatures et en étudiant leur répartition, pourrait donc mener une attaque statistique lui permettant de gagner de l'information sur la structure du code qui a permis de construire $f_{w,\mathbf{H}}$.

Notation Nous noterons $\mathbf{e}^{\text{unif}}$ la variable aléatoire distribuée comme :

$$\mathbf{e}^{\text{unif}} \sim \{\mathbf{e} \in \mathbb{F}_3^n : |\mathbf{e}| = w\}$$

Ainsi que $\mathbf{e}_V^{\text{unif}}$ et $\mathbf{e}_U^{\text{unif}}$ l'unique couple de variables aléatoires telles que :

$$\mathbf{e}^{\text{unif}} = (\mathbf{e}_U^{\text{unif}} \mid \mathbf{e}_U^{\text{unif}} + \mathbf{e}_V^{\text{unif}})$$

Objectif Notre objectif est que l'algorithme de décodage produise une solution uniformément distribuée parmi les antécédents de $f_{w,\mathbf{H}}$ quand $\mathbf{s} \in \mathbb{F}_q^{n-k}$ est distribuée uniformément.

Une solution pour y parvenir est de reprendre le décodage vu précédemment et d'y ajouter une méthode de rejet.

1.3.1 Méthode de rejet

Cela consiste, lors d'une expérience aléatoire, à filtrer certains résultats pour obtenir la loi de probabilité voulue. Par exemple, avec une pièce qui donne $\mathbb{P}(pile) = 1/3$ et $\mathbb{P}(face) = 2/3$, si on décide de refuser le résultat et de relancer la pièce avec 50% de probabilité à chaque fois qu'on obtiens *face*, la probabilité d'obtenir *face* diminuera et on simulera une loi uniforme avec cette pièce.

Nous allons donc mettre en place un algorithme du type de l'Algorithme 3.

Algorithme 3 DécodageUV($\mathbf{H}_U, \mathbf{H}_V, \mathbf{s}, w, \mathcal{D}_V, \mathcal{D}_U^t$)

1: **repeat**
2: $\quad\mathbf{e}_V \leftarrow$ DécodageV($\mathbf{H}_V, \mathbf{s}^V, \mathcal{D}_V$)
3: $\quad t \leftarrow |\mathbf{e}_V|$
4: **until** Condition 1 est vérifiée
5: **repeat**
6: $\quad k_{\neq 0} \hookleftarrow \mathcal{D}_U^t$
7: $\quad\mathbf{e}_U \leftarrow$ DécodageU($\mathbf{H}_U, \mathbf{s}^U, \mathbf{e}_V, k_{\neq 0}, w$)
8: $\quad\mathbf{e} \leftarrow (\mathbf{e}_U \mid \mathbf{e}_U + \mathbf{e}_V)$
9: **until** Condition 2 est vérifiée
10: **return** $\mathbf{e}$

La méthode de rejet est appliquée deux fois. L'algorithme fait en sorte grâce aux Conditions 1 et 2 que :

1. le vecteur $\mathbf{e}_V$ en entrée de DécodageV(.) à l'étape 7 a la même distribution que $\mathbf{e}_V^{\text{unif}}$.
2. le mot $\mathbf{e}_U$ passant la condition 2 à l'étape 9 conditionné à la valeur de $|\mathbf{e}_V|$ a la même distribution que $\mathbf{e}_U^{\text{unif}}$ conditionnée au vecteur $\mathbf{e}_V^{\text{unif}}$.

Dans ce rapport, nous allons nous restreindre à étudier le premier point. Pour cela, deux conditions à vérifier sont suffisantes :

1. $|\mathbf{e}_V| \sim |\mathbf{e}_V^{\text{unif}}|$
2. DécodageV doit vérifier une certaine condition, appelée uniformité en poids. Nous n'entrerons pas dans les détails de cette condition dans ce rapport, voir [Deb19, 5.3.1].

Ainsi, l'objectif se ramène à s'assurer que $|\mathbf{e}_V| \sim |\mathbf{e}_V^{\text{unif}}|$. Pour cela, on a l'Algorithme 4 (rand($[0,1]$) désigne un réel aléatoire choisit uniformément dans $[0,1]$).

Algorithme 4 RejetV($\mathbf{H}_V, \mathbf{s}, \mathcal{D}_V$)

1: **repeat**
2: $\quad\mathbf{e}_V \leftarrow$ DécodageV($\mathbf{H}_V, \mathbf{s}^V, \mathcal{D}_V$)
3: $\quad t \leftarrow |\mathbf{e}_V|$
4: **until** rand($[0,1]$) $\leq \mathbf{r}_V(t)$
5: **return** $\mathbf{e}_V$

Proposition 2. *Considérons pour $i, t \in [\![0, n/2]\!]$ et $s \in [\![0, t]\!]$*

$$q_1(i) \triangleq \mathbb{P}(|\mathbf{e}_V| = i), \quad q_1^{\text{unif}}(i) \triangleq \mathbb{P}(|\mathbf{e}_V^{\text{unif}}| = i)$$

$$\mathbf{r}_V(i) \triangleq \frac{1}{M_V^{rs}} \frac{q_1^{\text{unif}}(i)}{q_1(i)} \quad \text{avec } M_V^{rs} \triangleq \max_{0 \leq i \leq n/2} \frac{q_1^{\text{unif}}(i)}{q_1(i)}$$

Alors la sortie $\mathbf{e}_V$ de l'Algorithme 4 vérifie bien $|\mathbf{e}_V| \sim |\mathbf{e}_V^{unif}|$.

Démonstration de la Proposition 2. Voir [Deb19, 5.3] □

Proposition 3. *Le nombre de rejets moyen est donné par :*

$$M_V^{rs} - 1$$

Démonstration de la Proposition 3. Notons p la probabilité de quitter la boucle *while* dans l'Algorithme 3 en une itération.

$$p = \sum_{t=0}^{n/2} \mathbb{P}(|\mathbf{e}_V| = t)\mathbf{r}_V(t) = \frac{1}{M_V^{rs}} \sum_{t=0}^{n/2} q_1(t) \frac{q_1^{\text{unif}}(t)}{q_1(t)} = \frac{1}{M_V^{rs}}$$

Ainsi, d'après les propriétés sur les lois géométriques, le nombre moyen d'itération avant de quitter la boucle *while* sera de $\frac{1}{p} = M_V^{rs}$. Le nombre de rejet étant le nombre d'itération moins 1 (une seule itération signifie qu'il n'y a pas de rejet), on a le résultat. □

1.3.2 Objectif du stage

Objectif Il s'agit d'essayer de réduire le plus possible le taux de rejet moyen, donc de rapprocher le plus possible M_V^{rs} de 1. Si $M_V^{rs} - 1$ devient petit devant le paramètre de sécurité, alors on pourra se passer du rejet.

Pour cela, regardons les expressions explicites de q_1^{unif} et q_1.

Proposition 4. (Voir [Deb19, 5.3.2]) *Pour $i \in [\![0, n/2]\!]$, on a :*

$$q_1^{\text{unif}}(i) = \frac{\binom{n/2}{i}}{\binom{n}{w} 2^{w/2}} \sum_{p=0,\ 2|w+p}^{i} \binom{i}{p} \binom{n/2 - i}{\frac{w+p}{2} - i} 2^{\frac{3p}{2}} \tag{2}$$

Proposition 5. *Considérons, selon les notations de l'Algorithme 4, la distribution $\mathcal{D}_V$. Soit X_V la variable aléatoire telle que $X_V \sim \mathcal{D}_V$. Pour $i \in [\![0, n/2]\!]$, on a :*

$$q_1(i) = \frac{1}{3^{n/2 - k_v}} \sum_{t=0}^{i} \binom{n/2 - k_v}{i - t} 2^{i-t} \mathbb{P}(X_V = t) \tag{3}$$

Démonstrations de la Proposition 5. Il s'agit d'une conséquence directe de la Proposition 1. □

On remarque que X_V paramètre q_1, donc l'idée est de choisir le mieux possible X_V pour que q_1 se rapproche de q_1^{unif}. En effet, l'objectif est de minimiser $M_V^{rs} = \max_{0 \leq i \leq n/2} \frac{q_1^{\text{unif}}(i)}{q_1(i)}$. q_1^{unif} et q_1 étant des distributions de probabilité, leurs normes sont fixées à 1, et donc ce n'est qu'en rapprochant le plus possible ces distributions que l'on peut minimiser leur rapport.

Pour cela, notre seule marche de manœuvre est la distribution X_V. On voit dans l'Équation (3) qu'elle permet de modifier q_1. L'objet de notre travail a donc été de déterminer la meilleure distribution possible pour X_V afin rapprocher q_1 de q_1^{unif}, et donc de faire diminuer M_V^{rs}.

Nous avons travaillé dans cet objectif, et allons maintenant présenter nos résultats.

2 Résultats à paramètres fixés

Dans cette section, n, k, k_V et w sont fixés, suivant les paramètres standards de *Wave* qui sont, pour λ bits de sécurité, donnés par :

$$n = 66.34\lambda, \quad w = 0.9396n, \quad k_V = 0.4821\frac{n}{2} \quad \text{et} \quad \alpha = 0.574625 \tag{4}$$

Nous verrons plus loin ce que signifie le paramètre α. Tous les résultats numériques et graphes seront donnés selon les paramètres (4), pour $\lambda = 20$ bits de sécurité.

Introduisons les notations suivantes :

— X_V est la variable aléatoire que l'on choisit paramétrant la distribution q_1 (voir Équation (3))
— Y est une variable aléatoire suivant un schéma de Bernoulli de paramètres $2/3$ et $n/2 - k_V$

D'après la Proposition 1, la variable aléatoire émulée, suivant la distribution q_1, est la somme :

$$X_V + Y$$

On rappelle qu'on cherche à résoudre le problème suivant :

Problème 2. Minimisation du rejet

Minimiser M_V^{rs} en choisissant la variable aléatoire X_V de manière appropriée.

Pour cela, il faut que $X_V + Y$ suive le plus possible la distribution q_1^{unif}.

2.1 Recherche heuristique

2.1.1 Résumé de ce qui a été fait dans [Deb19, 5.3.4]

On note $\alpha > 0$ le réel tel que $(1-\alpha)k_V + \frac{2}{3}(n/2 - k_V) = \mathbb{E}(q_1^{\text{unif}})$

On cherche à rapprocher $X_V + Y$ de la distribution q_1^{unif}, on va donc choisir l'espérance de X_V de la façon suivante :

$$\mathbb{E}(X_V) = \mathbb{E}(q_1^{\text{unif}}) - \mathbb{E}(Y)$$

Or, $\mathbb{E}(Y) = \frac{2}{3}(n/2 - k_V)$, donc d'après la définition du paramètre α, on va prendre :

$$\mathbb{E}(X_V) = (1-\alpha)k_V$$

X_V constante Commençons par voir se qui se passe avec une distribution constante. On choisit X_V comme la variable aléatoire constante égale à $(1-\alpha)k_V$. On obtient par simulation la Figure 1. On rappelle que le taux de rejet moyen est donnée par $M_V^{rs} = \max_{0 \le i \le n/2} \frac{q_1^{\text{unif}}(i)}{q_1(i)}$. Or, on remarque par le calcul que dans les queues de distribution, $\frac{q_1^{\text{unif}}(i)}{q_1(i)}$ croît très vite, et le taux de rejet moyen devient très grand. On voudrait faire passer la courbe rouge au-dessus de la courbe bleue dans les queue de distribution. Pour cela, il faut augmenter la variance de X_V.

La distribution choisie pour ajouter de la variance à X_V est la suivante :

Définition 3. (Loi de Laplace tronquée et discrète (LLTD))
Soient μ, σ deux réels et a, b deux entiers. Une variable aléatoire est distribuée selon une loi de Laplace discrète et tronquée (LLTD) pour les paramètres μ, σ, a, b, ce que l'on notera $X \sim \text{Lap}(\mu, \sigma, a, b)$, si pour tout $i \in [\![a, b]\!]$,

$$\mathbb{P}(X = i) = \frac{e^{\frac{|i-\mu|}{\sigma}}}{N}$$

où N est un facteur de normalisation.

Ajout de variance à X_V On a $X_V \sim \text{Lap}((1-\alpha)k_V, \sigma, 0, k_V)$. Désormais, la courbe de X_e est au-dessus de celle de X_{unif} dans les queues de distribution, et le rejet a lieu à proximité de la moyenne. Voir la Figure 2. Le taux de rejet moyen est désormais raisonnable. Il peut atteindre l'ordre de 1% avec de bons paramètres.

Il a ensuite fallu que nous trouvions une distribution plus fine pour approcher q_1^{unif}.

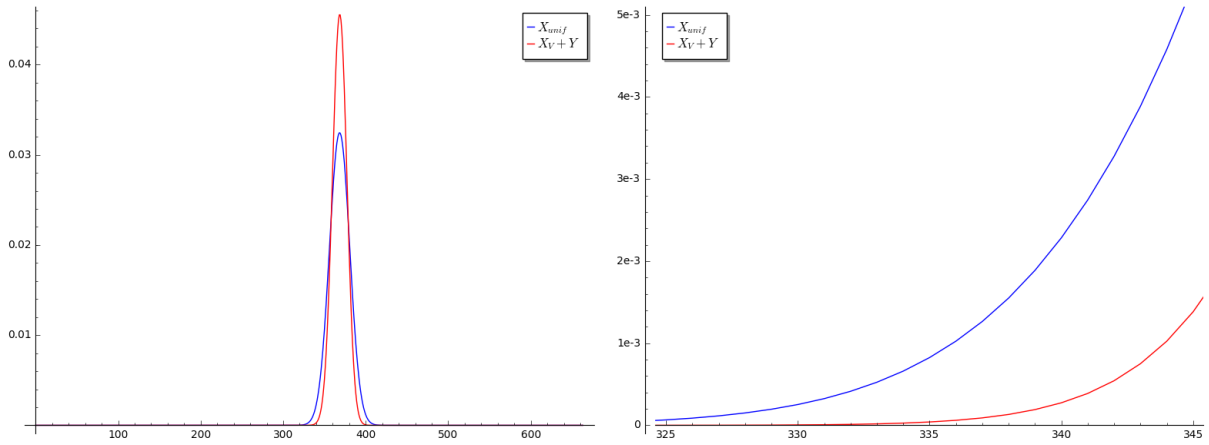


FIGURE 1 – Distributions q_1^{unif} en bleue et q_1 en rouge, pour $\mathcal{D}_V$ constante.

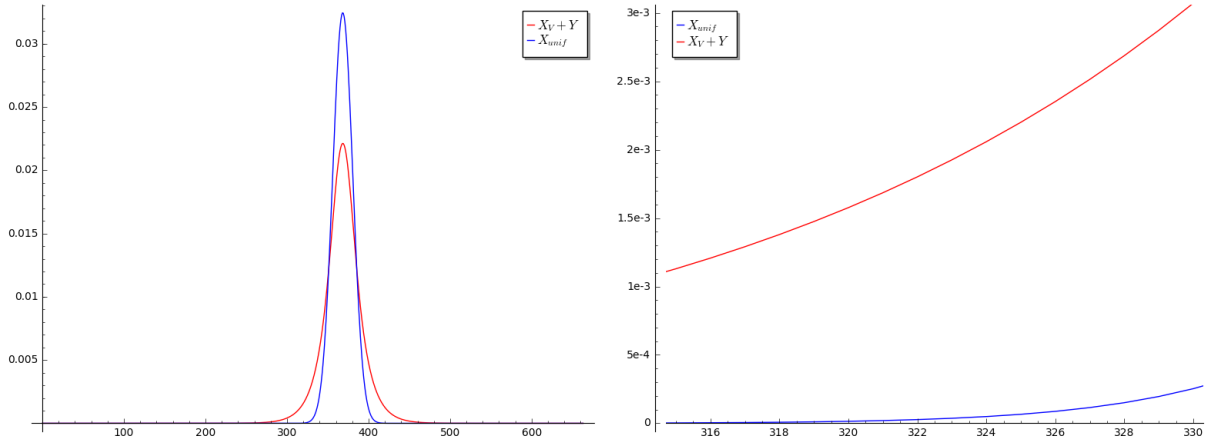


FIGURE 2 – Distributions q_1^{unif} en bleue et q_1 en rouge, pour $\mathcal{D}_V$ constante.

2.1.2 Les distributions en "Laplace par morceaux"

Pour améliorer le taux de rejet moyen, nous avons fait varier la variance des LLTD (le paramètre σ). Ainsi, il existe $p \geq 1$, des entiers $s_0 = 0 < s_1 < ... < s_p = k_V + 1$ et des réels $\sigma_0 \leq \sigma_1 \leq ... \leq \sigma_{p-1}$ tels que :

$$\mathbb{P}(X_V = i) = \frac{1}{N} \sum_{j=0}^{p-1} e^{\frac{|i-(1-\alpha)k_V|}{\sigma_j}} \mathbf{1}_{[\![s_i, s_{i+1}-1]\!]}$$

en notant $\mathbf{1}_I$ l'application valant 1 dans l'intervalle I, 0 en dehors. N est un facteur de normalisation.

En jouant "par tâtonnement" sur le nombre de subdivisions p et sur les variances σ_i, il a été possible de parvenir à faire baisser le taux de rejet moyen de plusieurs ordres de grandeurs. En atteignant par exemple, pour $\lambda = 20$, un taux de rejet moyen de l'ordre de 10^{-4}.

Il a ensuite fallu chercher des méthodes plus théoriques.

2.2 Transformée de Fourier

On notera $\mathcal{D}_V$ et $\mathcal{D}_Y$ les distributions respectives de X_V et Y. $*$ désigne le produit de convolution (la somme de variables aléatoires $X_V + Y$ suit donc la distribution $\mathcal{D}_V * \mathcal{D}_Y$).

Définition 4. *Ici, j désigne l'unité imaginaire, dont le carré vaut -1. On appelle transformée de Fourier*

discrète (TFD) d'une distribution à N termes $\mathcal{D}$, la distribution à N termes $TF(\mathcal{D})$, définie par :

$$\forall l \in [\![0, N-1]\!], \quad TF(\mathcal{D})_l \triangleq \sum_{n=0}^{N-1} \mathcal{D}(n) e^{-j2\pi \frac{nl}{N}}$$

La transformée de Fourier inverse d'une distribution $\mathcal{D}$ à N termes est donnée par :

$$\forall l \in [\![0, N-1]\!], \quad TF^{-1}(\mathcal{D})_l \triangleq \frac{1}{N} \sum_{n=0}^{N-1} \mathcal{D}(n) e^{j2\pi \frac{nl}{N}}$$

Propriétés de la TFD

— Pour toute distribution $\mathcal{D}$ définie sur $[\![0, N-1]\!]$, $TF^{-1}(TF(\mathcal{D})) = \mathcal{D}$.

— TF et TF^{-1} sont des opérateurs linéaires.

— Soient $\mathcal{D}_1$ et $\mathcal{D}_2$ deux distributions, $TF(\mathcal{D}_1 * \mathcal{D}_2) = TF(\mathcal{D}_1)TF(\mathcal{D}_2)$.

L'idée était d'essayer d'utiliser les propriétés de la transformée de Fourier, notamment sur le produit de convolution, pour déterminer $\mathcal{D}_V$ tel que $\mathcal{D}_V * \mathcal{D}_Y$ soit le plus proche possible de q_1^{unif}. On peut résoudre directement l'équation :

$$\begin{aligned} \mathcal{D}_V * \mathcal{D}_Y = q_1^{\text{unif}} &\iff TF(\mathcal{D}_V)TF(\mathcal{D}_Y) = TF(q_1^{\text{unif}}) \\ &\iff TF(\mathcal{D}_V) = \frac{TF(q_1^{\text{unif}})}{TF(\mathcal{D}_Y)} \\ &\iff \mathcal{D}_V = TF^{-1}\left(\frac{TF(q_1^{\text{unif}})}{TF(\mathcal{D}_Y)}\right) \end{aligned}$$

On considère ici $\mathcal{D}_V$ et $\mathcal{D}_Y$ comme étant définis sur $[\![0, n/2]\!]$, bien que les supports de $\mathcal{D}_V$ et $\mathcal{D}_Y$ soient respectivement $[\![0, k_V]\!]$ et $[\![0, n/2 - k_V]\!]$. Le résultat $\mathcal{D}_V$ est présenté sur la Figure 3.

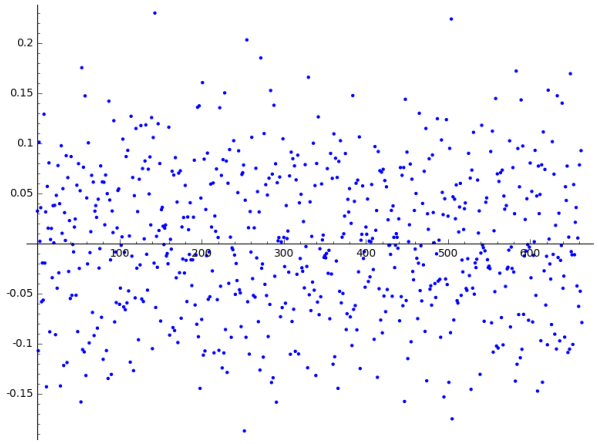


FIGURE 3 – Distribution $\mathcal{D}_V$, solution exacte de l'équation $\mathcal{D}_V * \mathcal{D}_Y = q_1^{\text{unif}}$

On remarque sur la Figure 3 que l'on n'obtient pas une distribution de probabilité, il y a des valeurs négatives (mais on a bien la somme des valeurs qui vaut 1). De plus, on a des valeurs non nulles en dehors de $[\![0, k_V]\!]$. Nous n'avons pas encore réussi à respecter les contraintes sur $\mathcal{D}_V$ en utilisant les transformées de Fourier. La méthode algorithmique suivante a été plus concluante.

2.3 Optimisation quadratique

Nous rappelons que nous cherchons à minimiser $M_V^{rs} = \max_{0 \leq i \leq n/2} \frac{q_1^{\text{unif}}(i)}{q_1(i)}$. Mais on peut aussi chercher à minimiser la distance statistique $\rho(q_1^{\text{unif}}, q_1)$. En effet, si $\rho(q_1^{\text{unif}}, q_1) \leq 2^{-\lambda}$, λ étant le nombre de bits de sécurité, alors les deux distributions deviennent indistinguables pour n'importe quel attaquant, et il n'y a plus besoin de rejet. Dans cette sous-section, nous cherchons à faire diminuer la distance statistique. Les expériences ayant été menée pour 20 bits de sécurité, nous faisons remarquer que $2^{-20} \simeq 10^{-6}$.

Ici, les distributions seront considérées comme des vecteurs ligne.

2.3.1 Présentation du problème

Minimiser $\rho(q_1^{\text{unif}}, q_1)$ revient à minimiser $||q_1^{\text{unif}} - q_1||_2$, où $||\cdot||_2$ désigne la norme 2. Nous allons développer $||q_1^{\text{unif}} - q_1||_2^2$. Pour alléger la notation, nous allons noter $X \stackrel{\triangle}{=} \mathcal{D}_V \in \mathbb{R}^{k_V}$. Remarquons préalablement que $q_1 = X\mathbf{A}$, $\mathbf{A} \in \mathbb{F}_3^{(k_V+1)\times(n+1)}$ où

$$\mathbf{A} = \begin{pmatrix} \mathcal{D}_Y(0) & \mathcal{D}_Y(1) & \cdots & \cdots & \cdots & \mathcal{D}_Y(n/2-k_V) & 0 & \cdots & 0 \\ 0 & \mathcal{D}_Y(0) & \mathcal{D}_Y(1) & \ddots & \ddots & \ddots & \mathcal{D}_Y(n/2-k_V) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathcal{D}_Y(0) & \mathcal{D}_Y(1) & \cdots & \cdots & \cdots & \mathcal{D}_Y(n/2-k_V) \end{pmatrix}$$

On a :

$$||q_1^{\text{unif}} - q_1||_2^2 = ||q_1^{\text{unif}} - X\mathbf{A}||_2^2 = ||X\mathbf{A}||_2^2 - 2 < X\mathbf{A}, q_1^{\text{unif}} > + ||q_1^{\text{unif}}||_2^2$$

Il faut donc minimiser :

$$||X\mathbf{A}||_2^2 - 2 < X\mathbf{A}, q_1^{\text{unif}} > \;\; = \;\; X\mathbf{A}\mathbf{A}^\intercal X^\intercal - 2q_1^{\text{unif}}\mathbf{A}^\intercal X^\intercal$$

sous les contraintes

$$\begin{cases} \forall i \in [\![0, k_V]\!], \quad X_i \geq 0 \\ \sum_{i=0}^{k_V} X_i = 1 \end{cases} \tag{5}$$

On est donc ramené au problème suivant :

Problème 3. (Problème d'optimisation quadratique sous contrainte)

Soit $m \in \mathbb{N}$, $\mathbf{C}$ une matrice carrée symétrique dans $\mathbb{R}$ de taille $m \times m$ et $\mathbf{v}$ une matrice colonne de taille m. Trouver :

$$\inf_{X \in K} \left(\frac{1}{2} X^\intercal \mathbf{C} X + \mathbf{v}^\intercal X \right)$$

où K est un sous-ensemble des matrices colonnes dans $\mathbb{R}$ de taille m.

En reprenant les notations de la définition du problème, nous avons dans notre cas $m = k_V + 1, \mathbf{C} = 2\mathbf{A}\mathbf{A}^\intercal$, $\mathbf{v} = 2\mathbf{A}q_1^{\text{unif}\intercal}$ et le sous-ensemble de matrices colonnes K est défini par (5). Il faut aussi remplacer X par sa transposée.

Résolution avec des programmes préinstallés Il existe en *Sage* et en *Matlab* des programmes préinstallés résolvant les instances du problème d'optimisation quadratique sous contrainte. Ceci a permis d'obtenir des paramètres aboutissant à une distance statistique de l'ordre de 10^{-3} en *Sage* et 10^{-4} en *Matlab*. J'ai cependant obtenu de meilleurs résultats en codant les algorithmes suivants.

2.3.2 Algorithme du gradient à pas fixe avec projection

L'algorithme du gradient sous contrainte est un algorithme classique d'optimisation (pas forcément quadratique) sous contrainte. Il tente donc de résoudre le problème suivant, qui est une version plus générale du Problème 3 :

Problème 4. *Trouver :*

$$\inf_{v \in K} J(v)$$

où J est une fonction α-convexe différentiable définie sur K, sous-ensemble convexe fermé non vide d'un espace de Hilbert réel V.

J est pour nous la fonction définie dans le Problème 3, V l'ensemble des matrices colonnes de longueur $k_V + 1$, et K est défini par (5). Notre problème remplit toutes les conditions nécessaires, notamment car notre matrice $\mathbf{C} = 2\mathbf{A}\mathbf{A}^\intercal$ est définie positive. Voir en Appendice pour plus de détails.

On note J' la différentielle de J. $\mathcal{P}_K$ est la projection orthogonale sur K, définie sur l'ensemble des matrices colonnes. Soit $\mu > 0$.

L'algorithme consiste à construire une suite u définie de la façon suivante :

$$\begin{cases} u^0 \in K \quad \text{quelconque} \\ u^{n+1} = \mathcal{P}_K(u^n - \mu J'(u^n)) \end{cases}$$

Proposition 6. *On suppose que J est α-convexe, J' est Lipschitzien de constante C. Alors si $0 < \mu < 2\alpha/C^2$, la suite u converge vers la solution du Problème 4.*

L'Algorithme 5 est une définition de l'algorithme dans le cas de l'optimisation quadratique. Il reprend les notations du Problème 3.

Algorithme 5 GradientQuadratique($\mathbf{C}, \mathbf{v}, K, X_0$)

Paramètre : m, $\mathcal{P}_K$ projection orthogonale sur K, $\varepsilon > 0$, $\mu > 0$

$X \leftarrow X_0$
$X' \leftarrow$ IterationGradient($\mathbf{C}, \mathbf{v}, \mathcal{P}_K, X, \mu$)
repeat
    $X \leftarrow X'$
    $X' \leftarrow$ IterationGradient($\mathbf{C}, \mathbf{v}, \mathcal{P}_K, X', \mu$)
until $||X' - X||_2 < \varepsilon$
return X'

function IterationGradient($\mathbf{C}, \mathbf{v}, \mathcal{P}_K, X, \mu$)

$\mathrm{d}X \leftarrow \mathbf{C}X + \mathbf{v}$
$Y \leftarrow \mathcal{P}_K(X - \mu \mathrm{d}X)$
return Y

Cependant, l'algorithme que l'on a effectivement mis en œuvre n'est pas exactement celui-ci. En effet, la projection orthogonale $\mathcal{P}_K$ de l'espace admissible K (donné dans (5)), est très compliquée à calculer. C'est pourquoi on a opté pour une projection approchée qui consiste à mettre tous les éléments négatifs de la distribution à 0, puis à normaliser la distribution en divisant chaque terme par la somme des termes.

Par ailleurs, déterminer la constante μ optimale nécessite de calculer précisément les valeurs propres de la matrice $\mathbf{C} = 2\mathbf{A}\mathbf{A}^\intercal \in \mathbb{F}_3^{(k_V+1)\times(k_V+1)}$, ce qui je n'ai pas pu faire en temps raisonnable avec les algorithmes de calcul de valeurs propres de *Sage* ou de *Matlab*. Nous avons donc choisi un $\mu > 0$ suffisamment petit pour que la suite des distributions successives semble converger. Plus précisément :

En codant en *Sage* IterationGradient avec $\mathcal{P}'_K$ la version approchée de $\mathcal{P}_K$ et $\mu = 0.2$, nous obtenons une suite (X_n) définie par $X_{n+1} =$ IterationGradient($\mathbf{C}, \mathbf{v}, \mathcal{P}'_K, X, \mu$) avec X_0 quelconque telle que :

— $\left(\frac{1}{2} {X_n}^\intercal \mathbf{C} X_n + \mathbf{v}^\intercal X_n\right)$ est strictement décroissante, mais décroît de moins en moins vite.

— On arrive à une distance statistique $\rho(q_1^{\text{unif}}, q_1)$ de l'ordre de $10^{-7<2^{-20}}$. Ce qui est mieux que tout ce qu'on a eu auparavant, mais insuffisant pour se passer du rejet avec 64 bits. On ne parvient pas à faire tourner l'algorithme suffisamment longtemps pour aller en-dessous de cet ordre, si jamais c'est possible.

Nous avons utilisé un algorithme approché en raison de la difficulté de la projection dur K. L'algorithme suivant est un algorithme exact.

2.3.3 Algorithme d'Uzawa

L'algorithme d'Uzawa est un algorithme permettant également de résoudre les problèmes d'optimisation sous contrainte, justement prévu pour le cas où la projection sur l'ensemble des solutions admissibles est compliquée à calculer.

Malheureusement, en raison soit d'imprécisions dans les calculs, soit de la nature du problème, on ne parvient pas à obtenir une meilleur distance statistique que de l'ordre de 10^{-4}, ce qui n'est pas mieux que lors de la méthode précédente.

Voilà pour les résultats à n fixé. Nous nous sommes intéressé à un autre problème, de nature asymptotique cette fois, qui n'avait pas encore été résolu jusque là. C'est l'objet de la section suivante.

3 Résultat asymptotique

Le problème consiste à voir, lorsqu'on fait tendre la taille des paramètres vers l'infini, si on peut trouver une suite de distributions qui permette un taux de rejet moyen qui augmente de façon polynomiale en la taille des paramètres. Cela signifierait que quelque soit le niveau de sécurité, il existe une instanciation qui fait fonctionner la méthode de rejet. Contrairement à [DST19] où il est montré que la méthode de rejet fonctionne pour un jeu de paramètres donné. Comme $n = 66.34\lambda$, faire tendre n vers l'infini revient à faire tendre le paramètre de sécurité vers l'infini. Il s'agit donc de voir si on pourra continuer à faire fonctionner *Wave* si le nombre de bits de sécurité augmente.

Soit $n \in \mathbb{N}$. On va faire tendre n vers l'infini. On notera :

$$w = \omega n, \quad k_V = R_V \frac{n}{2}$$

On reste selon les paramètres de (4) que l'on rappelle ici :

$$n = 66.34\lambda, \quad \omega = 0.9396, \quad R_V = 0.4821$$

A n fixé, on note aussi :

— X_V^n la variable aléatoire à valeur dans $[\![0, k_V]\!]$ que l'on choisi.
— Y^n la variable aléatoire suivant un schéma de Bernoulli de paramètres n et $2/3$.

i est une variable entière comprise entre 0 et k_V. Comme k_V tend avec n vers l'infini, on introduit la variable asymptotique η définie telle que $i = \eta\frac{n}{2}$ où $0 \leq \eta \leq 1$.

Nous allons résoudre le problème suivant :

Problème 5. *Montrer qu'il existe une suite $(X_V^n)_{n\geq 0}$ telle que le taux de rejet moyen est asymptotiquement polynomial en n quand n tend vers l'infini.*

Nous allons montrer qu'une distribution constante convient, et permet de résoudre le problème.

X_V^n est donc distribuée sur $[\![0, k_V]\!]$ de la manière suivante :

$$\mathbb{P}(X_V^n = t) = M(n)$$

où

$$M(n) \triangleq \frac{1}{k_V + 1} \sim \frac{2}{R_V n} \tag{6}$$

Pour montrer que cette distribution résout le Problème 5, nous allons démontrer la Proposition suivante, en supposant vraie la Conjecture 1 énoncée page 19 :

Proposition 7. *En supposant vraie la Conjecture 1, il existe un polynôme P tel que pour n suffisamment grand :*

$$\forall \eta \in [0,1], \quad \frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)} \leq P(n)$$

Pour cela, nous allons montrer que lorsque n tend vers l'infini, le quotient $\frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)}$ tend vers 0 quand η se trouve dans les intervalles $\left[0, \frac{2}{3}(1-R_V)\right]$ et $\left[\frac{2}{3}(1-R_V)+R_V, 1\right]$, et est polynomial quand η se trouve dans l'intervalle $\left[\frac{2}{3}(1-R_V), \frac{2}{3}(1-R_V)+R_V\right]$.

Nous allons maintenant étudier séparément ces intervalles. L'étude sur $\left[\frac{2}{3}(1-R_V)+R_V, 1\right]$ étant symétrique à celle sur $\left[0, \frac{2}{3}(1-R_V)\right]$, nous la mettrons en appendice.

3.1 Intervalle $\left[\frac{2}{3}(1-R_V), \frac{2}{3}(1-R_V)+R_V\right]$

L'intérêt de se concentrer sur cet intervalle est que pour $\eta \in \left[\frac{2}{3}(1-R_V), \frac{2}{3}(1-R_V)+R_V\right]$,

$$q_1\left(\left\lfloor \eta \frac{n}{2} \right\rfloor\right) \geq \mathbb{P}\left(Y^n = \left\lfloor \left(\frac{2}{3}(1-R_V)\right) \frac{n}{2} \right\rfloor\right) \mathbb{P}\left(X_V^n = \left\lfloor \left(\eta - \frac{2}{3}(1-R_V)\right) \frac{n}{2} \right\rfloor\right)$$

Y^n suivant une loi binomiale de moyenne $\frac{2}{3}\left((1-R_V)\frac{n}{2}\right)$, elle est diminue de façon polynomiale en n en cette valeur. On obtient donc le résultat suivant :

Proposition 8. *Il existe une constante strictement positive μ, telle que pour tout $\eta \in \left[\frac{2}{3}(1-R_V), \frac{2}{3}(1-R_V)+R_V\right]$ et n suffisamment grand on a :*

$$\frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)} \leq \mu n \sqrt{n}$$

Démonstration. Soit $\eta \in \left[\frac{2}{3}(1-R_V), \frac{2}{3}(1-R_V)+R_V\right]$. On rappelle :

$$q_1\left(\left\lfloor \eta \frac{n}{2} \right\rfloor\right) = \frac{1}{3^{n/2-k_V}} \sum_{t=0}^{\lfloor \eta \frac{n}{2} \rfloor} \binom{n/2-k_V}{\lfloor \eta \frac{n}{2} \rfloor - t} 2^{\lfloor \eta \frac{n}{2} \rfloor - t} \mathbb{P}(X_V^n = t)$$

On va se concentrer sur le terme dans la somme où $\lfloor \eta n \rfloor - t = \frac{2}{3}(1-R_V)$, qui est toujours atteignable lorsque $\eta \in \left[\frac{2}{3}(1-R_V), \frac{2}{3}(1-R_V)+R_V\right]$, puis appliquer la formule (1). On a :

$$q_1\left(\left\lfloor \eta \frac{n}{2} \right\rfloor\right) \geq \frac{1}{3^{n/2-k_V}} \binom{n/2-k_V}{\frac{2}{3}(n/2-k_V)} 2^{\frac{2}{3}(n/2-k_V)} \mathbb{P}(X_V^n = t) \tag{7}$$

$$\frac{1}{3^{n/2-k_V}} \binom{n/2-k_V}{\frac{2}{3}(n/2-k_V)} 2^{\frac{2}{3}(n/2-k_V)} \mathbb{P}(X_V^n = t) \sim \frac{3}{\sqrt{2\pi n(1-R_V)}} M(n) \sim \frac{3}{R_V} \sqrt{\frac{2}{\pi(1-R_V)}} \frac{1}{n\sqrt{n}} \tag{8}$$

Par ailleurs, q_1^{unif} étant une distribution de probabilité, on a :

$$q_1^{\text{unif}}\left(\left\lfloor \eta \frac{n}{2} \right\rfloor\right) \leq 1 \tag{9}$$

En combinant (7), (8) et (9), on a le résultat. □

3.2 Intervalle $\left[0, \frac{2}{3}(1-R_V)\right]$

Pour l'étude de cet intervalle, nous devons dès maintenant introduire les fonctions suivantes :

— Pour $x \in]0,1[$:

$$f^{\text{unif}}(x) \stackrel{\triangle}{=} \frac{h_3(x)}{2} - h_3(\omega) + \frac{1}{2} \max_{\max(0,x-\omega) \leq y \leq \min(x/2, 1-\omega)} \left\{ x h_3\left(\frac{2y}{x}\right) + (1-x) h_3\left(\frac{\omega+y-x}{1-x}\right) \right\}$$

— Pour $x \in \left]0, \frac{2}{3}(1-R_V)\right]$:

$$g_1(x) \stackrel{\triangle}{=} f^{\text{unif}}(x) - (1/2 - R_V/2)\left(h_3\left(\frac{x}{1-R_V}\right) - 1\right)$$

— Pour $x \in \left[\frac{2}{3}(1-R_V)+R_V, 1\right]$:

$$g_2(x) \stackrel{\triangle}{=} f^{\text{unif}}(x) - (1/2 - R_V/2)\left(h_3\left(\frac{x-R_V}{1-R_V}\right) - 1\right)$$

f^{unif} sert pour l'étude asymptotique de q_1^{unif}, comme l'indique la Proposition suivante :

Proposition 9. *Il existe une constante $C > 0$ telle que pour n suffisamment grand on a :*

$$\forall \eta \in]0,1[, \quad q_1^{\text{unif}}\left(\left\lfloor \eta \frac{n}{2} \right\rfloor\right) \leq Cn \times 3^{n \cdot f^{\text{unif}}(\eta)}$$

Démonstration. Utiliser l'équation (1) dans (2) la formule de q_1^{unif}. □

g_1 servira pour l'étude de $\frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)}$ pour $\eta \in \left]0, \frac{2}{3}(1-R_V)\right]$, comme le laisse voir la Proposition suivante :

Proposition 10. *Pour tout $0 < a < b \leq \frac{2}{3}(1-R_V)$ il existe une constante $C > 0$ telle que pour n suffisamment grand,*

$$\forall \eta \in [a,b], \quad q_1\left(\left\lfloor \eta \frac{n}{2} \right\rfloor\right) \geq \frac{C}{\sqrt{n}} 3^{(1/2-R_V/2)\left(h_3\left(\frac{\eta}{1-R_V}\right)-1\right)n + \log(M(n))}$$

Démonstration. Comme on a $\eta \leq 1 - R_V$, on peut remarquer que :

$$q_1\left(\left\lfloor \eta \frac{n}{2} \right\rfloor\right) \geq \mathbb{P}\left(Y^n = \left\lfloor \eta \frac{n}{2} \right\rfloor\right) \mathbb{P}(X_V^n = 0)$$

Puis en appliquant l'Equation (1) à Y, on obtient le résultat. □

g_2 servira dans l'intervalle $\left[\frac{2}{3}(1-R_V)+R_V, 1\right[$, voir l'Appendice.

Nous allons devoir faire la conjecture suivante, vérifiée numériquement (voir Figure 4) :

Conjecture 1. *g_1 et g_2 sont strictement négatives sur leurs intervalles de définition respectifs.*

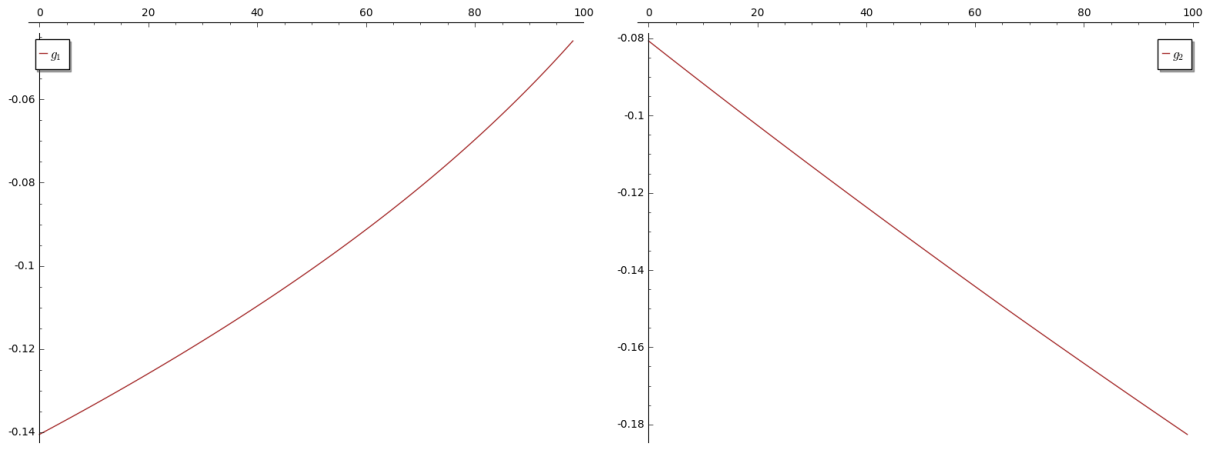


FIGURE 4 – Fonctions $x \to g_1(\frac{2x}{300}(1-R_V))$ et $x \to g_2(\frac{2}{3}(1-R_V)+R_V+\frac{x}{100}(1/3-R_V/3))$

Pour étudier $\left[0, \frac{2}{3}(1-R_V)\right]$, on va séparer cet intervalle en deux, $[0,\varepsilon]$ et $\left[\varepsilon, \frac{2}{3}(1-R_V)\right]$.

3.2.1 Intervalle $[0, \varepsilon]$

Proposition 11. *Pour $D > 0$ suffisamment petit,*

$$i \mapsto q_1^{\text{unif}}(i) \quad \text{et} \quad i \mapsto q_1(i)$$

sont deux fonctions croissantes sur $[\![0, \lfloor Dn \rfloor]\!]$.

Proposition 12. *Il existe $\varepsilon > 0$ tel que pour n suffisamment grand on a,*

$$\forall \eta \in [0, \varepsilon], \quad \frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)} \leq 1.$$

Démonstration. Par la Proposition 11 on a pour x suffisamment petit :

$$\forall \eta \in [0, x], \quad \frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)} \leq \frac{q_1^{\text{unif}}(\lfloor xn \rfloor)}{q_1(0)}$$

$q_1(0) = \frac{M(n)}{3^{n/2-k_V}}$, donc avec la Proposition 9 on déduit l'existence d'une constante $C > 0$ telle que :

$$\forall \eta \in [0, x], \quad \frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)} \leq C \times 3^{n \cdot \left(f^{\text{unif}}(x) - \frac{\log(M(n))}{n} + (1/2 - R_V/2) \right)} \tag{10}$$

Nous allons montrer qu'il existe $\varepsilon > 0$ et $\beta < 0$ tels que

$$\forall \eta \in [0, \varepsilon], \quad f^{\text{unif}}(\eta) - \frac{\log(M(n))}{n} + (1/2 - R_V/2) < \beta < 0. \tag{11}$$

De cette manière, en remplaçant x par ε dans (10), on aura, pour $\eta \in [0, \varepsilon]$, $\frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)}$ qui tend vers 0, et on aura terminé la preuve.

On a avec les paramètres standards de (4) :

$$-\frac{h_3(\omega)}{2} + (1/2 - R_V/2) < 0$$

$M(n)$ tend vers 0 de façon polynomiale en n (Équation (6)), donc $\frac{\log(M(n))}{n}$ tend vers 0, et donc il existe un réel $\beta < 0$ tel que :

$$-\frac{h_3(\omega)}{2} + (1/2 - R_V/2) - \frac{\log(M(n))}{n} < \beta < 0$$

On a $f^{\text{unif}}(\eta) \underset{\eta \to 0^+}{\to} -\frac{h_3(\omega)}{2}$, donc en utilisant la continuité de f^{unif}, on peut choisir $\varepsilon > 0$ tel que (11) est vérifiée. □

3.2.2 Intervalle $\left[\varepsilon, \frac{2}{3}(1 - R_V)\right]$

Proposition 13. *Sous l'hypothèse de la Conjecture 1, pour tout $\eta \in \left[\varepsilon, \frac{2}{3}(1 - R_V)\right]$ il existe une constante $C > 0$ ainsi que $\gamma < 0$ telles que pour n suffisamment grand on a :*

$$\frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)} \leq n\sqrt{n} C \times 3^{\gamma n}$$

Démonstration. D'après les Propositions 9 et 10, on a l'existence de $C > 0$ telle que :

$$\forall \eta \in \left[\varepsilon, \frac{2}{3}(1 - R_V)\right], \quad \frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)} \leq n\sqrt{n} C \times 3^{\left(g_1(\eta) - \frac{\log(M(n))}{n}\right) n}$$

Or, comme $M(n)$ tend vers 0 de façon polynomiale, on a $g_1(\eta) - \frac{\log(M(n))}{n}$ qui tend vers $g_1(\eta) < 0$ (Conjecture 1), d'où le résultat. □

Proposition 14. *Sous l'hypothèse de la Conjecture 1 :*

$$\forall \eta \in \left[\varepsilon, \frac{2}{3}(1 - R_V)\right], \quad \frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)} \leq 1.$$

Démonstration. Conséquence immédiate de la Proposition 13. □

3.3 Conclusion

Ainsi, en combinant les résultats des Propositions 8, 12 et 14, on démontre la Proposition 7, sous réserve de la validité de la Conjecture 1.

Références

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779) :505–510, 2019.

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012*, LNCS. Springer, 2012.

[BM17] Leif Both and Alexander May. Optimizing BJMM with Nearest Neighbors : Full Decoding in $2^{2/21n}$ and McEliece Security. In *WCC Workshop on Coding and Cryptography*, September 2017.

[Deb19] Thomas Debris-Alazard. *Cryptographie fondée sur les codes : nouvelles approches pour constructions et preuves ; contribution en cryptanalyse. (Code-based Cryptography : New Approaches for Design and Proof ; Contribution to Cryptanalysis)*. PhD thesis, Pierre and Marie Curie University, Paris, France, 2019.

[DST19] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave : A new family of trapdoor one-way preimage sampleable functions based on codes. In *Advances in Cryptology - ASIACRYPT 2019*, LNCS, Kobe, Japan, December 2019.

[Dum91] Ilya Dumer. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, pages 50–52, Moscow, 1991.

[LB88] Pil J. Lee and Ernest F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In *Advances in Cryptology - EUROCRYPT'88*, volume 330 of *LNCS*, pages 275–280. Springer, 1988.

[McE78] Robert J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $O(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, 2011.

[MO15] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 203–228. Springer, 2015.

[Ste88] Jacques Stern. A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *LNCS*, pages 106–113. Springer, 1988.

A Résultat asymptotique

A.1 Intervalle $\left[\frac{2}{3}(1-R_V)+R_V, 1\right]$

On va séparer cet intervalle en deux, $[1-\varepsilon, 1]$ et $\left[\frac{2}{3}(1-R_V)+R_V, 1-\varepsilon\right]$.

A.1.1 Intervalle $[1-\varepsilon, 1]$

Proposition 15. *Pour $D>0$ suffisamment petit,*

$$i \mapsto q_1^{\text{unif}}(i) \quad \textit{et} \quad i \mapsto q_1(i)$$

sont deux fonctions décroissantes sur $[\![n/2 - \lfloor Dn \rfloor, n/2]\!]$.

Proposition 16. *Il existe $\varepsilon > 0$ tel que pour n suffisamment grand on a,*

$$\forall \eta \in [1-\varepsilon, 1], \quad \frac{q_1^{\text{unif}}(\lfloor \eta n \rfloor)}{q_1(\lfloor \eta n \rfloor)} \leq 1.$$

Démonstration. On a avec les paramètres standards de (4) :

$$-h_3(\omega) + \frac{h_3(2(1-\omega))}{2} + \frac{\log_3(2)}{2} + (1/2 - R_V/2)(1-\log_3(2)) < 0$$

$M(n)$ tend vers 0 de façon polynomiale en n (Equation (6)), donc il existe $\beta < 0$ tel que pour n suffisamment grand on a :

$$-h_3(\omega) + \frac{h_3(2(1-\omega))}{2} + \frac{\log_3(2)}{2} + (1/2 - R_V/2)(1-\log_3(2)) - \frac{\log(M(n))}{n} < \beta < 0$$

On a $f^{\text{unif}}(\eta) \underset{\eta \to 1/2^-}{\to} -h_3(\omega) + \frac{h_3(2(1-\omega))}{2} + \frac{\log_3(2)}{2}$, donc en utilisant la continuité de f^{unif}, on peut choisir $\varepsilon > 0$ tel que :

$$\forall \eta \in [1-\varepsilon, 1], \quad f^{\text{unif}}(\eta) + (1/2 - R_V/2)(1-\log_3(2)) - \frac{\log(M(n))}{n} < \beta < 0. \tag{12}$$

Maintenant, d'après la Proposition 15 on a que :

$$\forall \eta \in [1-\varepsilon, 1], \quad \frac{q_1^{\text{unif}}(\lfloor \eta \frac{n}{2} \rfloor)}{q_1(\lfloor \eta \frac{n}{2} \rfloor)} \leq \frac{q_1^{\text{unif}}(\lfloor \frac{n}{2}(1-\varepsilon) \rfloor)}{q_1(n/2)}$$

$q_1(n/2) = \left(\frac{2}{3}\right)^{n/2-k_V} M(n)$, donc avec la Proposition 9 on déduit l'existence d'une constante $C > 0$ telle que :

$$\begin{aligned} \forall \eta \in [1-\varepsilon, 1], \quad \frac{q_1^{\text{unif}}(\lfloor \eta n \rfloor)}{q_1(\lfloor \eta n \rfloor)} &\leq Cn \times 3^{n \cdot \left(f^{\text{unif}}(1/2-\varepsilon) - \frac{\log(M(n))}{n} + (1/2 - R_V/2)(1-\log_3(2))\right)} \\ &\underset{n \to +\infty}{\longrightarrow} 0 \end{aligned}$$

où la dernière ligne vient de (12). □ □

A.1.2 Intervalle $\left[\frac{2}{3}(1-R_V)+R_V, 1-\varepsilon\right]$

Proposition 17. *Pour tout $\frac{2}{3}(1-R_V)+R_V \leq a < b < 1$, il existe une constante $C > 0$ telle que pour n suffisamment grand,*

$$\forall \eta \in [a, b], \quad q_1\left(\left\lfloor \eta \frac{n}{2} \right\rfloor\right) \geq \frac{C}{\sqrt{n}} 3^{(1/2 - R_V/2)\left(h_3\left(\frac{\eta - R_V}{1-R_V}\right) - 1\right) n + \log(M(n))}$$

Démonstration. Comme on a $\eta \geq R_V$, on peut remarquer que :

$$q_1\left(\left\lfloor \eta\frac{n}{2}\right\rfloor\right) \geq \mathbb{P}\left(Y^n = \left\lfloor \eta\frac{n}{2}\right\rfloor - k_V\right)\mathbb{P}(X_V^n = k_V)$$

Puis en appliquant l'Equation (1) à Y, on obtient le résultat. □

Proposition 18. *Sous l'hypothèse de la Conjecture 1, pour tout $\eta \in \left[\frac{2}{3}(1-R_V)+R_V, 1-\varepsilon\right]$, il existe une constante $C > 0$ ainsi que $\gamma < 0$ telles que pour n suffisamment grand nous avons :*

$$\frac{q_1^{\text{unif}}(\lfloor \eta\frac{n}{2}\rfloor)}{q_1(\lfloor \eta\frac{n}{2}\rfloor)} \leq n\sqrt{n}C \times 3^{\gamma n}$$

Démonstration. D'après les Propositions 9 et 17, il existe $C > 0$ tel que :

$$\frac{q_1^{\text{unif}}(\lfloor \eta\frac{n}{2}\rfloor)}{q_1(\lfloor \eta\frac{n}{2}\rfloor)} \leq n\sqrt{n}C \times 3^{\left(g_2(\eta)-\frac{\log(M(n))}{n}\right)n}$$

Or, comme $M(n)$ tend vers 0 de façon polynomiale, on a $g_2(\eta) - \frac{\log(M(n))}{n}$ qui tend vers $g_2(\eta) < 0$ (Conjecture 1), d'où le résultat. □

Proposition 19. *Sous l'hypothèse de la Conjecture 1 :*

$$\forall \eta \in \left[\frac{2}{3}(1-R_V)+R_V, 1-\varepsilon\right], \quad \frac{q_1^{\text{unif}}(\lfloor \eta\frac{n}{2}\rfloor)}{q_1(\lfloor \eta\frac{n}{2}\rfloor)} \leq 1.$$

Démonstration. Conséquence immédiate de la Proposition 18. □

On a donc :

Proposition 20. *Sous l'hypothèse de la Conjecture 1, pour tout $\eta \in \left[\frac{2}{3}(1-R_V)+R_V, 1\right]$, on a pour n suffisamment grand :*

$$\frac{q_1^{\text{unif}}(\lfloor \eta\frac{n}{2}\rfloor)}{q_1(\lfloor \eta\frac{n}{2}\rfloor)} \leq 1$$

Démonstration. En combinant la Proposition 16 et la Proposition 19. □